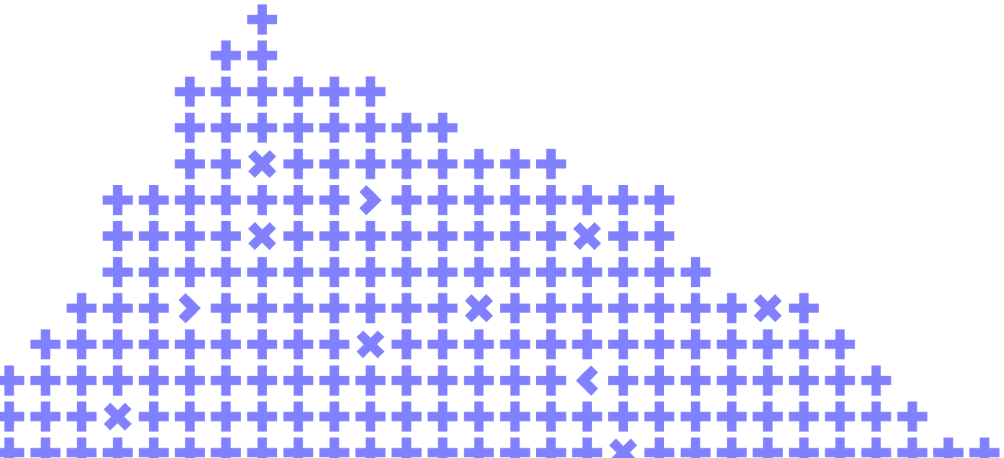
# Push notifications in RuStore: how we built an alternative transport to replace Google Firebase

Kirill Alekseev



# About me



# About me

- Got the bachelor's degree and the master’s degree of the MSU Faculty of Computational Mathematics and Cybernetics

# About me



- Got the bachelor's degree and the master’s degree of the MSU Faculty of Computational Mathematics and Cybernetics
- At VK I manage several backend development teams in Mail.ru Mail Service

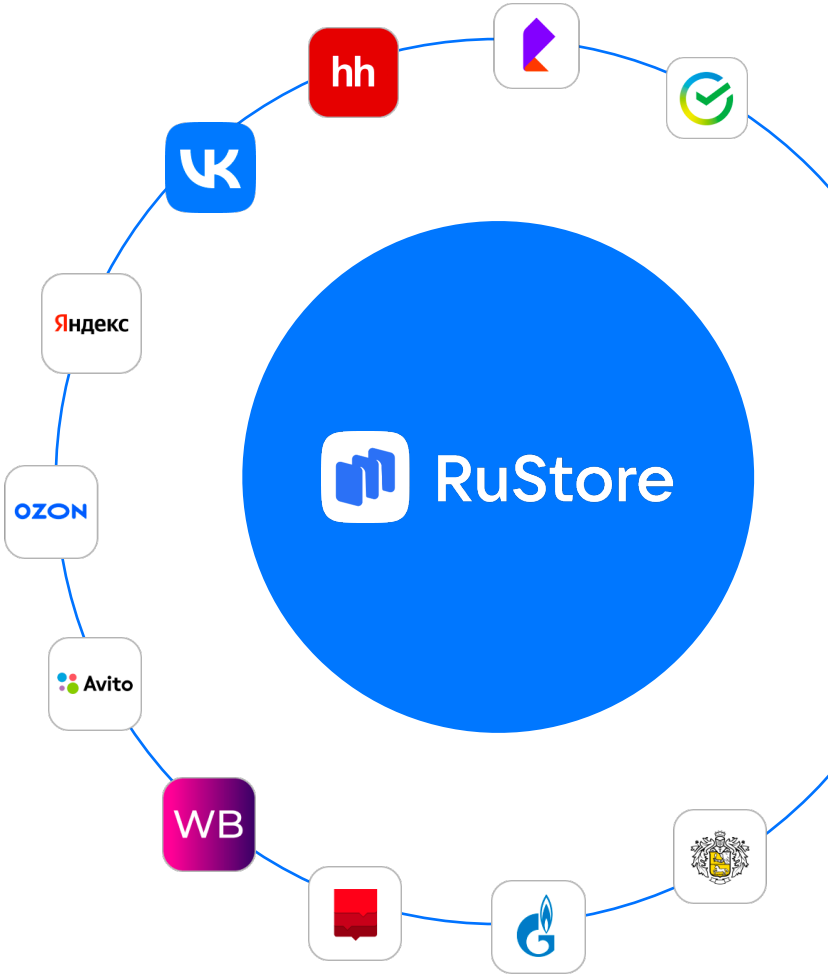
# About me



- Got the bachelor's degree and the master’s degree of the MSU Faculty of Computational Mathematics and Cybernetics
- At VK I manage several backend development teams in Mail.ru Mail Service
- Playing guitar and planning sprints at "Dela Povazhnee"

# RuStore is being developed with the contribution of the largest Russian IT companies.

Thank you for your contacts!



# Table of Contents


# Table of Contents



# Table of Contents



# Table of Contents



# Table of Contents



# Table of Contents



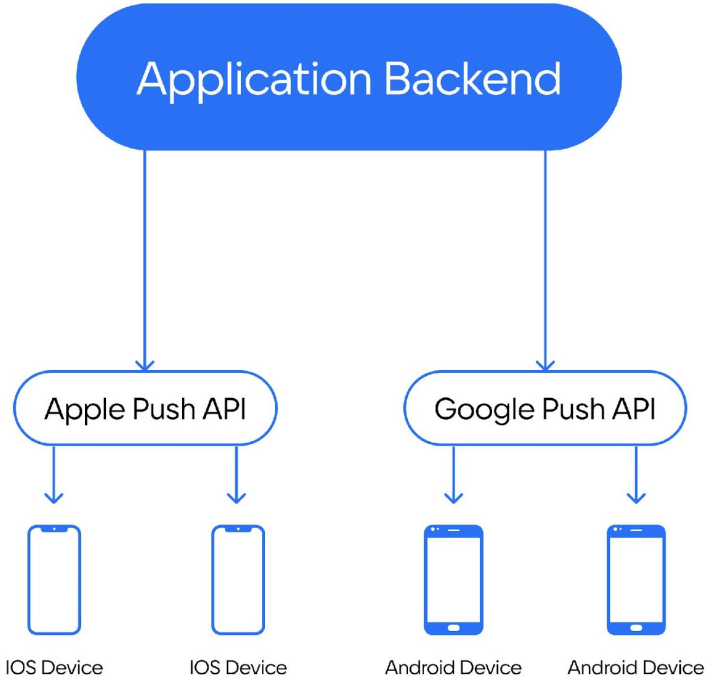
# Table of Contents



RuStore

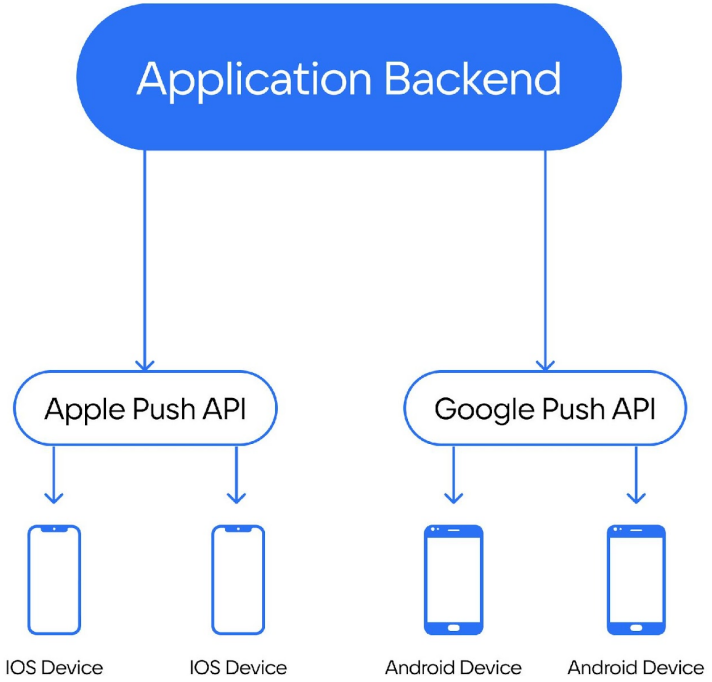
# Introduction

# Why build our own transport of push notifications



# Why build our own transport of push notifications

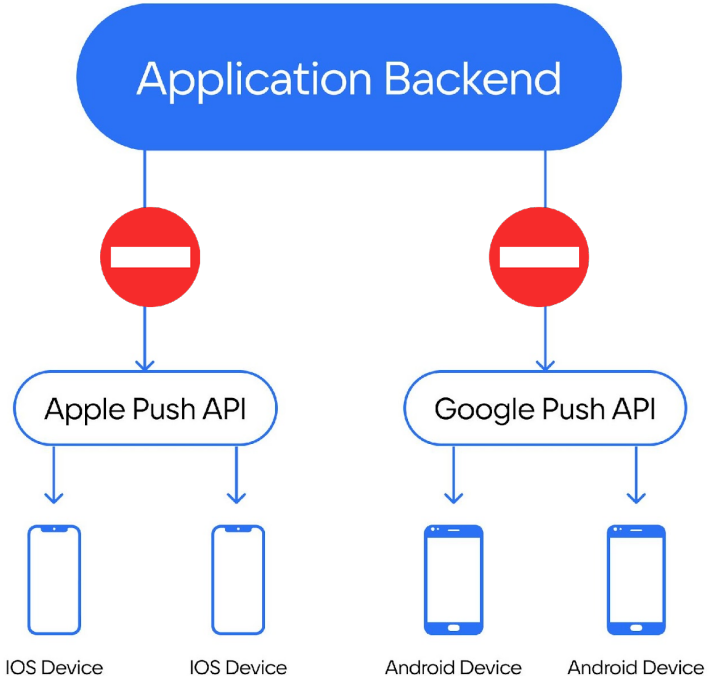
Push notifications are sent
via Google/Apple/Huawei API



# Why build our own transport of push notifications

Push notifications are sent
via Google/Apple/Huawei API

If Google decides
that we no longer need
push notifications, then
we no longer have it



# Requirements for the service

7

# Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications

# Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device

# Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)

# Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)
- Service must be reliable

# Requirements for the service



- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)
- Service must be reliable
- Service must be secure

# Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)
- Service must be reliable
- Service must be secure
- Easy integration into the current scheme of working with push notifications

# What any push notification looks like

# What any push notification looks like



- Push token identifies the application on a specific device

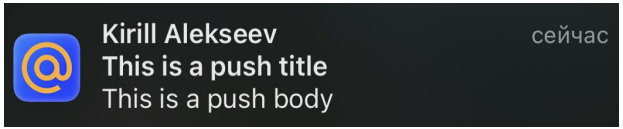
```
{
  "token": "PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD",
  "title": "Kirill Alekseev\nThis is a push title",
  "body": "This is a push body",
  "some_other_useful_data": "42"
}
```

# What any push notification looks like



- Push token identifies the application on a specific device
- In a push notification, you can send an arbitrary payload and process it client-side

```
{
  "token": "PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD",
  "title": "Kirill Alekseev\nThis is a push title",
  "body": "This is a push body",
  "some_other_useful_data": "42"
}
```

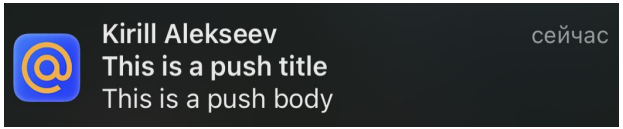


# What any push notification looks like



- Push token identifies the application on a specific device
- In a push notification, you can send an arbitrary payload and process it client-side
- Built-in push notification transports use internal OS **mechanisms that are inaccessible to us**

```
{
  "token": "PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD",
  "title": "Kirill Alekseev\nThis is a push title",
  "body": "This is a push body",
  "some_other_useful_data": "42"
}
```

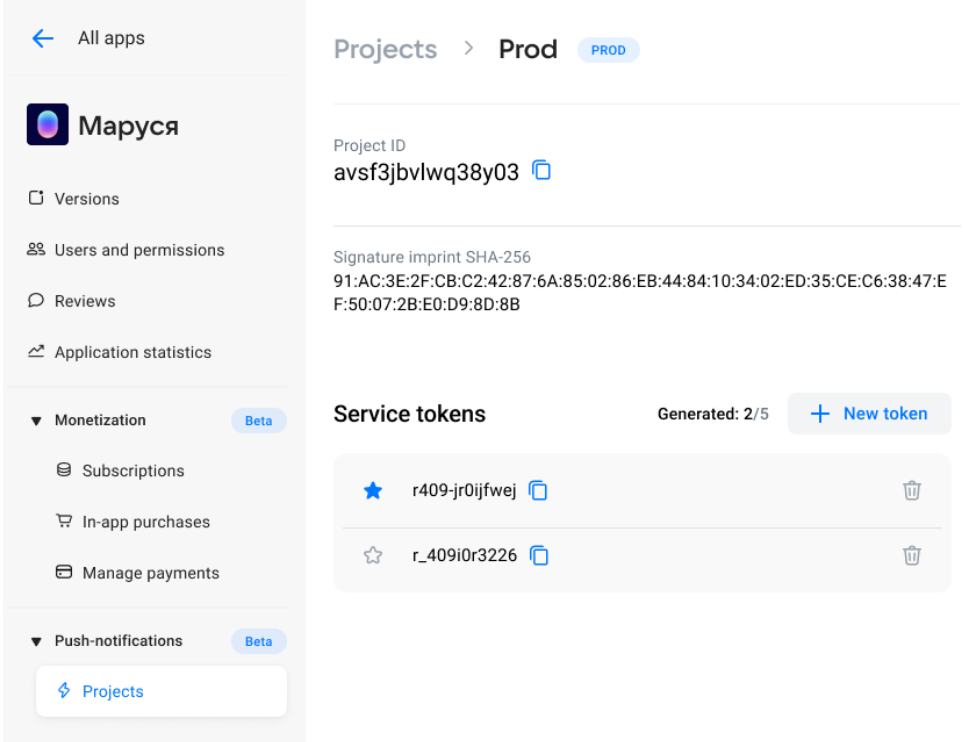


RuStore

# Overview of the service’s public part

# A web-interface to manage push notifications of application in RuStore

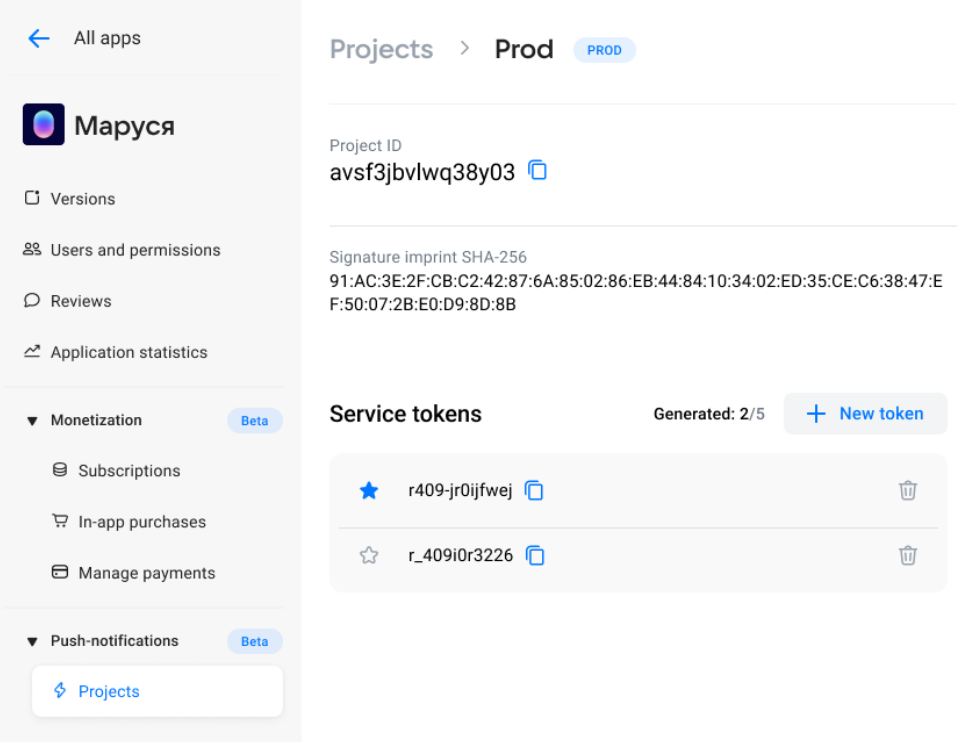
10




High Load ++ Armenia

# A web-interface to manage push notifications of application in RuStore

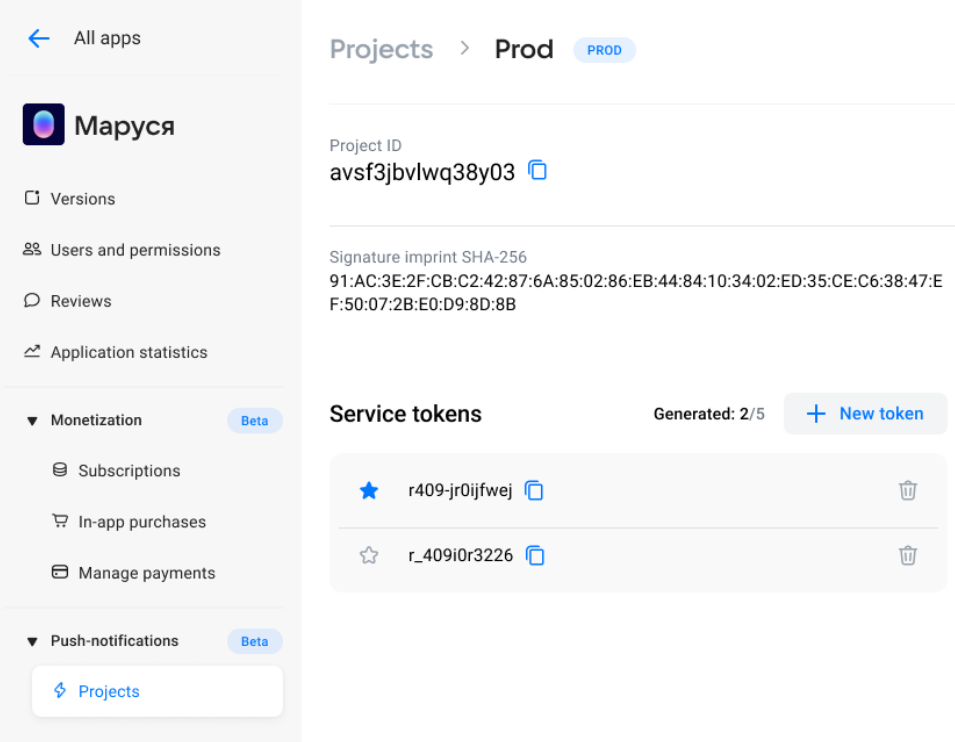





Any application is identified with the project ID

# A web-interface to manage push notifications of application in RuStore

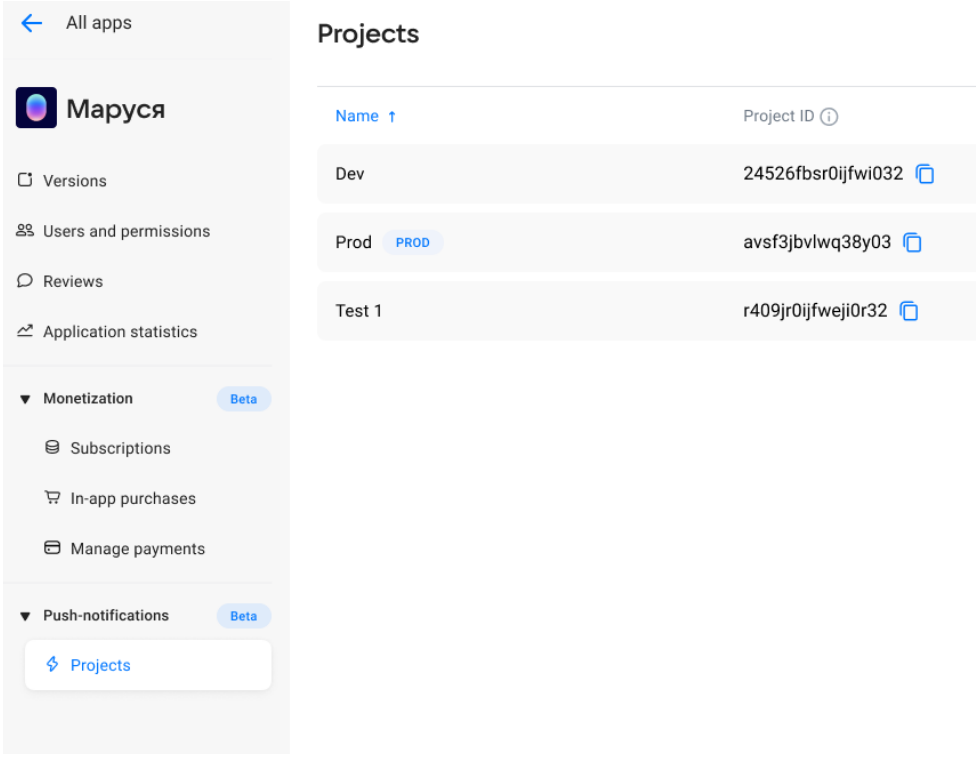
10



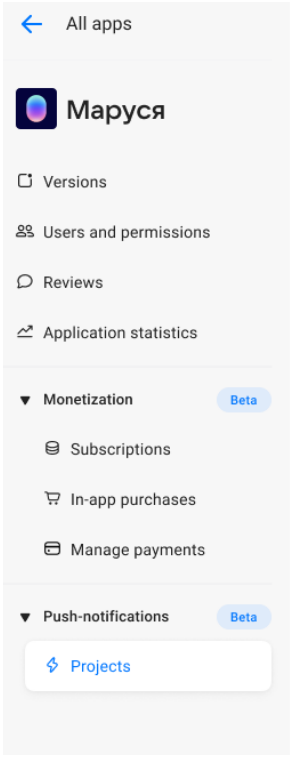

Any application is identified with the project ID

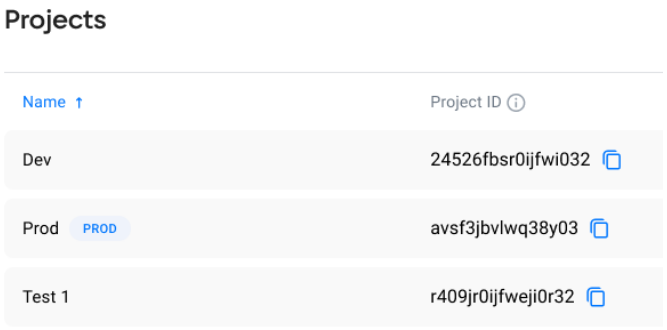
Service tokens authorize sending a push notification **and are sensitive**

# A web-interface to manage push notifications of application in RuStore



# A web-interface to manage push notifications of application in RuStore




You can isolate the Dev and the Prod service tokens by linking several push projects to one application

# SDK for push service in RuStore

12

# SDK for push service in RuStore



- We try to be a drop-in replacement for FCM SDK

# SDK for push service in RuStore



- We try to be a drop-in replacement for FCM SDK
- The getToken method registers and gives the application a new push token

# SDK for push service in RuStore



- We try to be a drop-in replacement for FCM SDK
- The getToken method registers and gives the application a new push token
- The deleteToken method invalidates an existing token

# SDK for push service in RuStore



- We try to be a drop-in replacement for FCM SDK
- The getToken method registers and gives the application a new push token
- The deleteToken method invalidates an existing token
- You can define callbacks onMessageReceived, onNewToken, onDeletedMessages

# RuStore Push Service API Methods



```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \
    --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \
    --data-raw '{"message":{
      "token":"PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD",
      "android":{ "notification":{ "title":"Hello, world!"} }
    }}'
```

# RuStore Push Service API Methods



- We try to be a drop-in replacement for the FCM API

```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \
    --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \
    --data-raw '{"message":{
      "token":"PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD",
      "android":{ "notification":{ "title":"Hello, world!"} }
    }}'
```

# RuStore Push Service API Methods



- We try to be a drop-in replacement for the FCM API
- We support both notification and data pushes, but so far a limited set of fields

```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \
    --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \
    --data-raw '{"message":{
      "token":"PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD",
      "android":{ "notification":{ "title":"Hello, world!"} }
    }}'
```


# Internals of our SDK

# Main
# SDK components

# Main
# SDK components



- There are two SDKs:
"client" and "host"

# Main
# SDK components



- There are two SDKs: "client" and "host"
- "Host" is built into RuStore and collects **all push notifications for a given device**

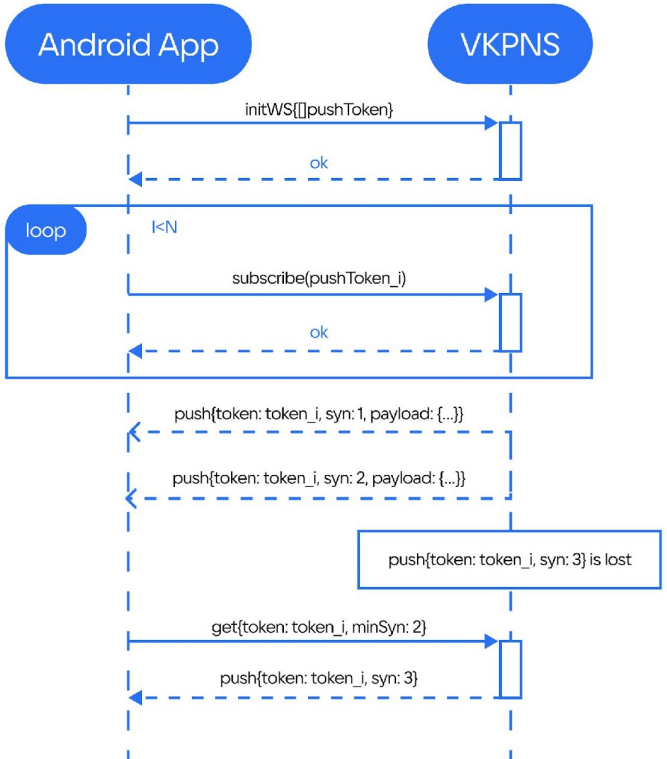
# Main
# SDK components



- There are two SDKs: "client" and "host"
- "Host" is built into RuStore and collects **all push notifications for a given device**
- "Client" must be integrated by the application developer (similar to the Firebase SDK)

# How SDK receives pushes

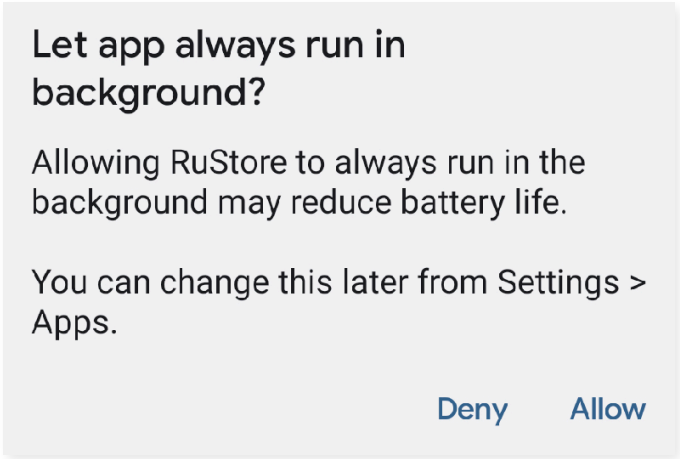
Most notifications come instantly via WebSocket, but HTTP polling is also performed periodically to deliver the lost ones.



# Fetching notifications in the background



Background instant delivery of push notifications requires a permission from the user



RuStore

# Push service backend architecture details

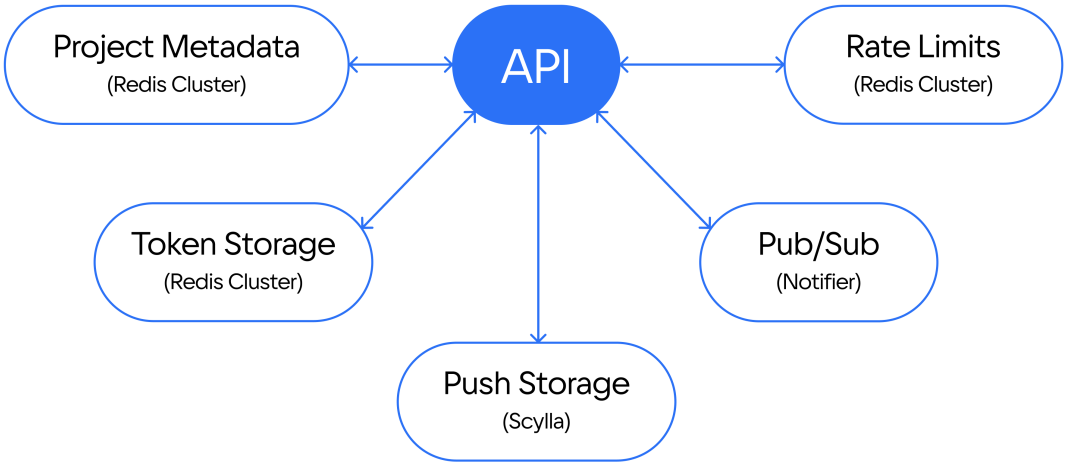
# Main components of the backend



Golang API in k8s

Redis Cluster

Scylla



# Push token storage overview

# Push token storage overview



Stored in Redis Cluster

```
hgetall token:{8B83XYXrmCYB9xFOmozIk-uZN}
1) "project_id"
2) "9TLrisVIsNfEc0DjE4clbJYsyyG9ACs4"
3) "syn"
4) "145"
7) "syn_stored"
8) “145"
```

# Push token storage overview



Stored in Redis Cluster

SYN — monotonically increasing push counter

```
hgetall token:{8B83XYXrmCYB9xFOmozIk-uZN}
1) "project_id"
2) "9TLrisVIsNfEc0DjE4clbJYsyyG9ACs4"
3) "syn"
4) "145"
7) "syn_stored"
8) “145"
```

# Push token storage overview



Stored in Redis Cluster

SYN — monotonically increasing push counter

Requires less than 400 MB for 1 million tokens

```
hgetall token:{8B83XYXrmCYB9xFOmozIk-uZN}
1) "project_id"
2) "9TLrisVIsNfEc0DjE4clbJYsyyG9ACs4"
3) "syn"
4) "145"
7) "syn_stored"
8) “145"
```

# Push storage overview

# Push storage overview



Stored in Scylla
on disk

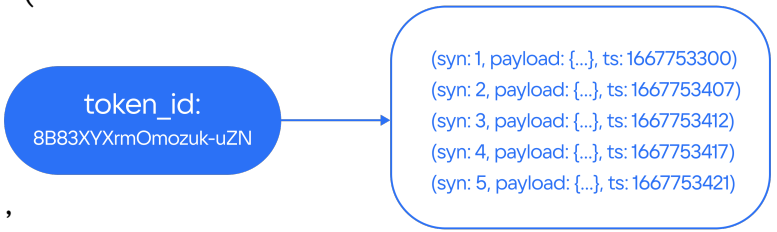
```
CREATE TABLE vkpns.push (
    token_id text,
    syn bigint,
    payload text,
    timestamp timestamp,
    PRIMARY KEY (token_id, syn)
) WITH CLUSTERING ORDER BY (syn ASC)
    AND bloom_filter_fp_chance = 0.01
    AND compaction = {'class': 'SizeTieredCompactionStrategy'}
    AND compression = {'sstable_compression': ‘LZ4Compressor'};
```

# Push storage overview



Stored in Scylla
on disk

```
CREATE TABLE vkpns.push (
    token_id text,
    syn bigint,
    payload text,
    timestamp timestamp,
    PRIMARY KEY (token_id, syn)
) WITH CLUSTERING ORDER BY (syn ASC)
    AND bloom_filter_fp_chance = 0.01
    AND compaction = {'class': 'SizeTieredCompactionStrategy'}
    AND compression = {'sstable_compression': 'LZ4Compressor'};
```



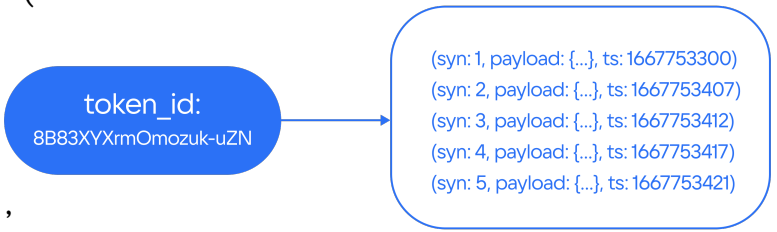
# Push storage overview



Stored in Scylla
on disk

Allows efficient reading
by token+syn_range

```
CREATE TABLE vkpns.push (
    token_id text,
    syn bigint,
    payload text,
    timestamp timestamp,
    PRIMARY KEY (token_id, syn)
) WITH CLUSTERING ORDER BY (syn ASC)
    AND bloom_filter_fp_chance = 0.01
    AND compaction = {'class': 'SizeTieredCompactionStrategy'}
    AND compression = {'sstable_compression': 'LZ4Compressor'};
```



# Push storage overview

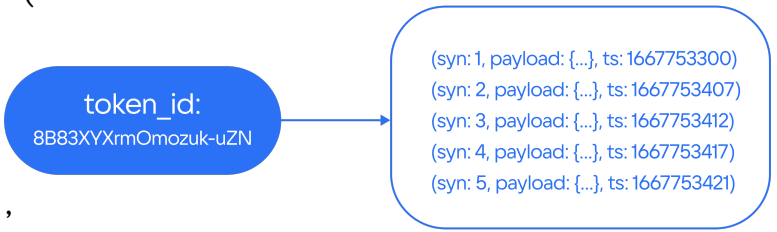


Stored in Scylla
on disk

Allows efficient reading
by token+syn_range

LWW (Last Write Wins)
saves us from duplicate
push notifications

```
CREATE TABLE vkpns.push (
    token_id text,
    syn bigint,
    payload text,
    timestamp timestamp,
    PRIMARY KEY (token_id, syn)
) WITH CLUSTERING ORDER BY (syn ASC)
    AND bloom_filter_fp_chance = 0.01
    AND compaction = {'class': 'SizeTieredCompactionStrategy'}
    AND compression = {'sstable_compression': 'LZ4Compressor'};
```



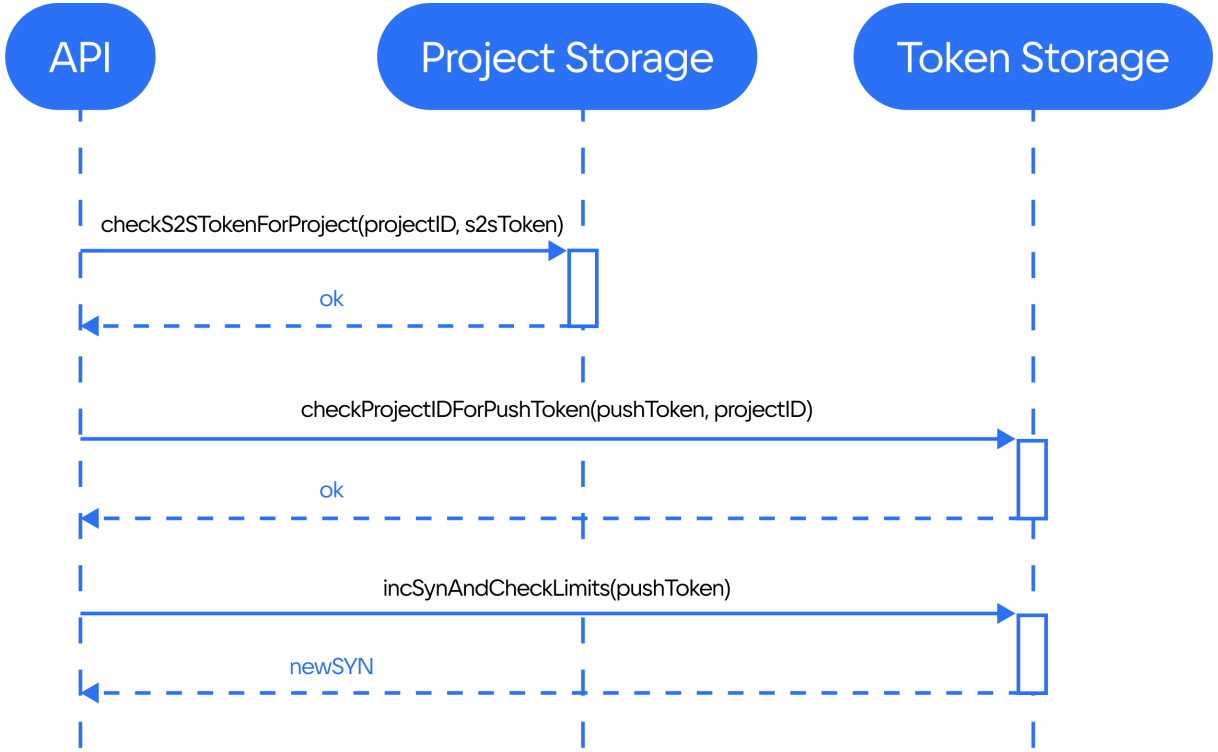
# API logic for sending a push notification



```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \
    --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \
    --data-raw '{"message":{
      "token":"PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD",
      "android":{ "notification":{ "title":"Hello, world!"} }
    }}'
```

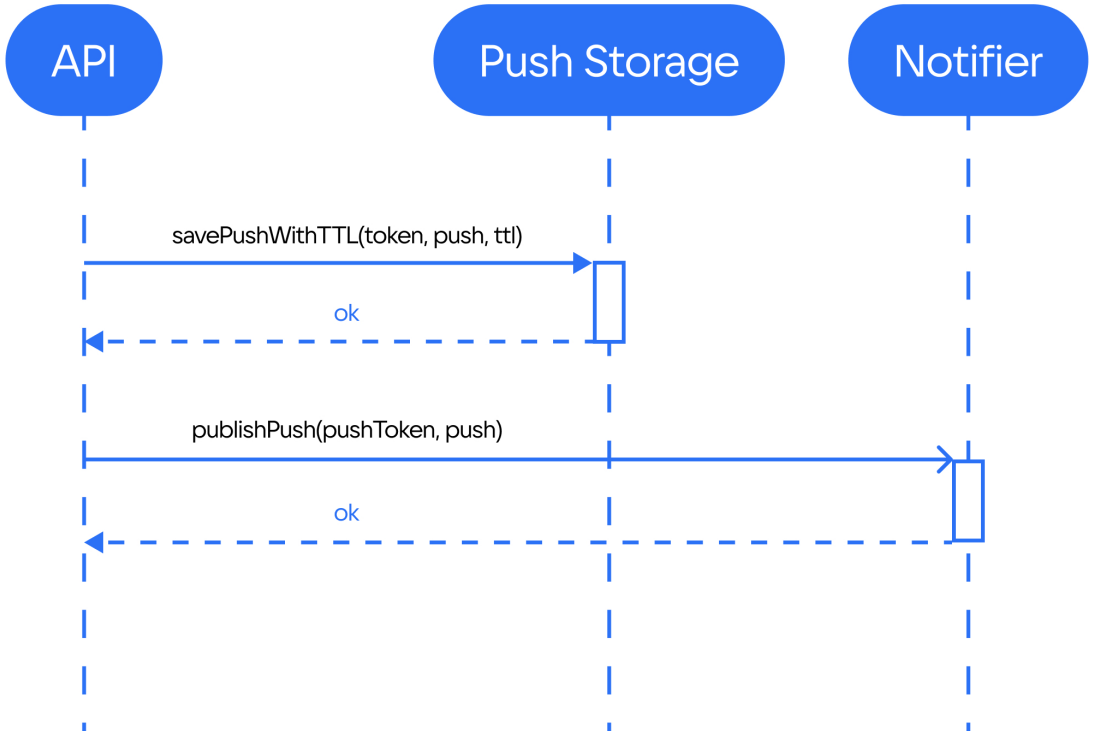
# API logic for sending a push notification





# API logic for sending a push notification



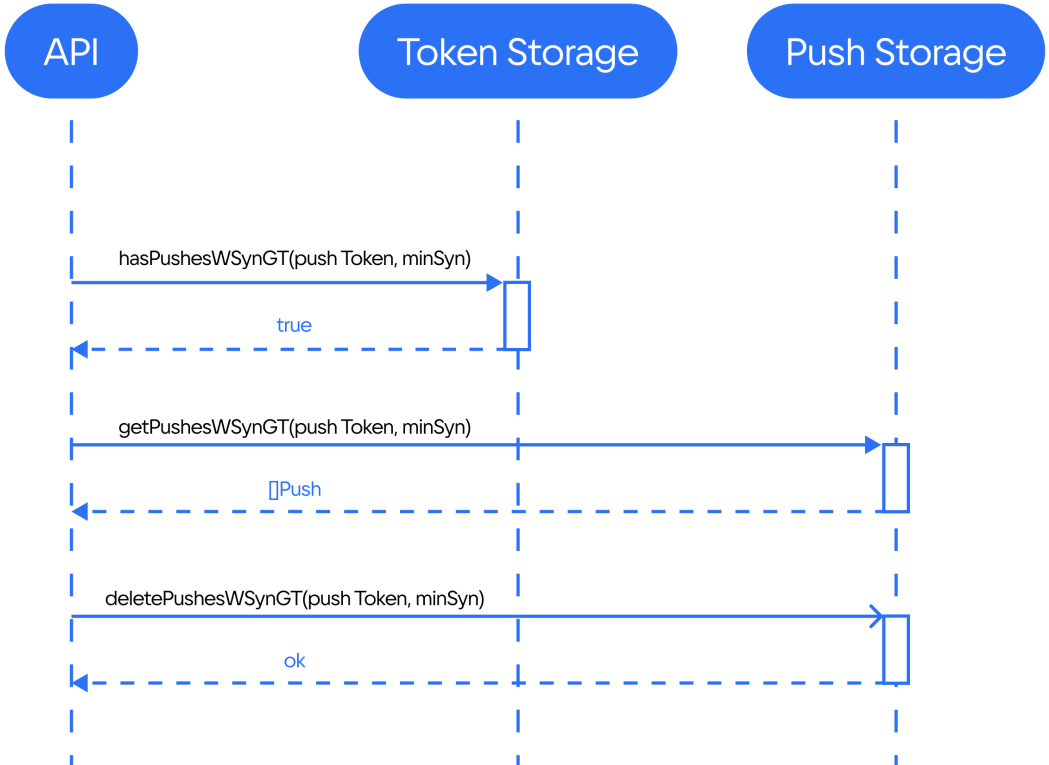


# API logic for receiving a push notification



```
curl 'https://vkpns.rustore.ru/v1/projects/messages:get' -H'Content-Type: application/
json' \
    --data-raw '{ "requests": [
        {
            "project_id": "admlkdas", "min_syn": 12, "limit": 10,
            "token": "PFGUGOBfCPZe3UOFU3UflV8qPGUddEBD"
        }
    ]}'
```

# API logic for receiving a push notification



# How not to overload the storage with stale push tokens

27

# How not to overload the storage with stale push tokens



- Reasons why stale tokens appear:

# How not to overload the storage with stale push tokens



- Reasons why stale tokens appear:

  app has been removed

# How not to overload the storage with stale push tokens



- Reasons why stale tokens appear:

  app has been removed

  phone got lost

# How not to overload the storage with stale push tokens



Reasons why stale tokens appear:

app has been removed

phone got lost

???

# How not to overload the storage with stale push tokens



- Reasons why stale tokens appear:

  app has been removed

  phone got lost

  ???

- All push tokens have a TTL in the token storage

# How not to overload the storage with stale push tokens



- Reasons why stale tokens appear:

  app has been removed

  phone got lost

  ???

- All push tokens have a TTL in the token storage

- Automatically update the lifetime of the token, if the client comes for pushes

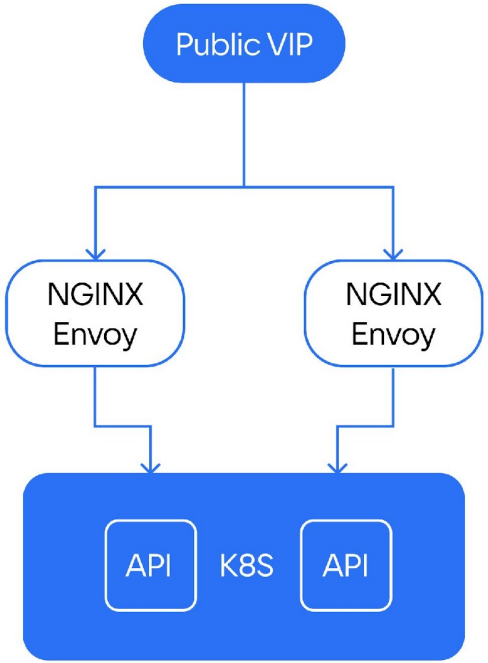

# Why we won't be down (at least for too long)

# Fault tolerance at the API level

API lives in Kubernetes

Kubernetes lives in 5 DCs

Pod healthchecks configured in Envoy



# Redis Cluster for Storage of push tokens

# Redis Cluster for Storage of push tokens



- Scales horizontally thanks to:

# Redis Cluster for Storage of push tokens



- Scales horizontally thanks to:

  - automatic resharding

# Redis Cluster for Storage of push tokens



- Scales horizontally thanks to:
  - automatic resharding
  - routing of read queries to replicas

# Redis Cluster for Storage of push tokens



- Scales horizontally thanks to:
  - automatic resharding
  - routing of read queries to replicas
- Fault-tolerant in case of a node/datacenter failure thanks to:

# Redis Cluster for Storage of push tokens



- Scales horizontally thanks to:

  - automatic resharding

  - routing of read queries to replicas

- Fault-tolerant in case of a node/datacenter failure thanks to:

  - automatic failover (with short downtime)

# Redis Cluster for Storage of push tokens



Scales horizontally thanks to:

- automatic resharding
- routing of read queries to replicas

Fault-tolerant in case of a node/datacenter failure thanks to:

- automatic failover (with short downtime)
- retries of read queries to another replica

# Scylla for Storage of pushes

# Scylla for Storage of pushes



- Scales horizontally with automatic resharding

# Scylla for Storage of pushes



- Scales horizontally with automatic resharding
- Fault-tolerant in case of a node/datacenter failure thanks to the "no master" architecture

# Scylla for Storage of pushes



- Scales horizontally with automatic resharding
- Fault-tolerant in case of a node/datacenter failure thanks to the "no master" architecture
- With no downtime!

# Scylla for Storage of pushes



- Scales horizontally with automatic resharding
- Fault-tolerant in case of a node/datacenter failure thanks to the "no master" architecture
- With no downtime!

~~Million~~

# 10 million WebSockets and Go

# ~~Million~~
# 10 million WebSockets and Go

- The service has been successfully operated in Mail.ru Mail Service for five years

# ~~Million~~
# 10 million WebSockets and Go

- The service has been successfully operated in Mail.ru Mail Service for five years
- Resistant to one DC failure

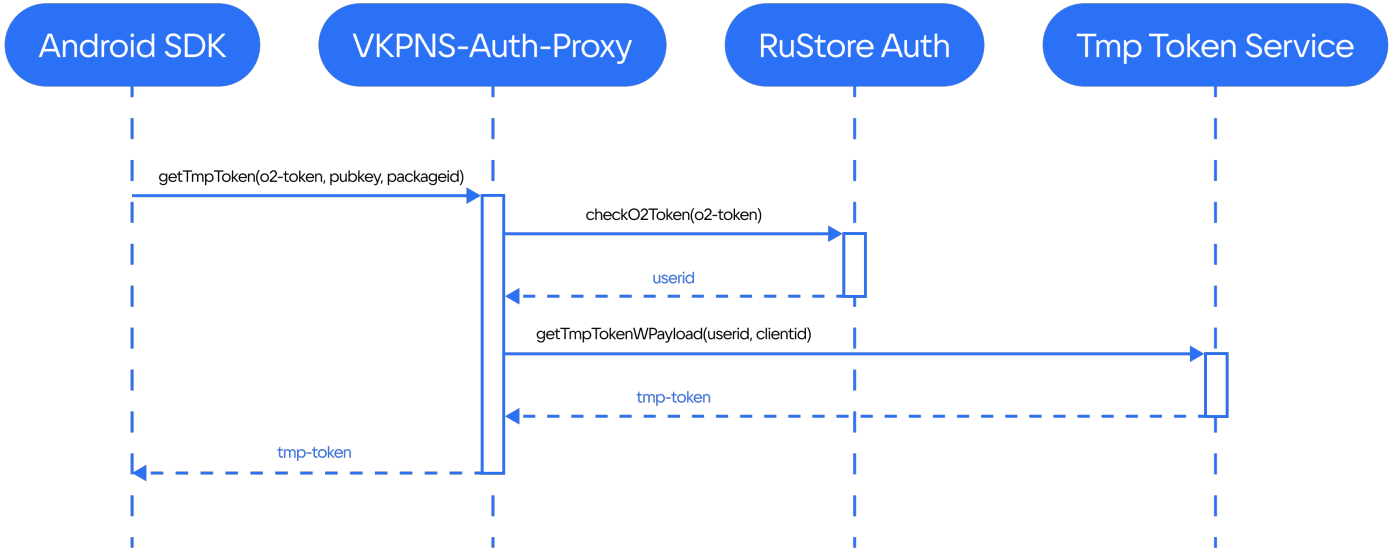
# ~~Million~~
# 10 million WebSockets and Go

32

- The service has been successfully operated in Mail.ru Mail Service for five years
- Resistant to one DC failure

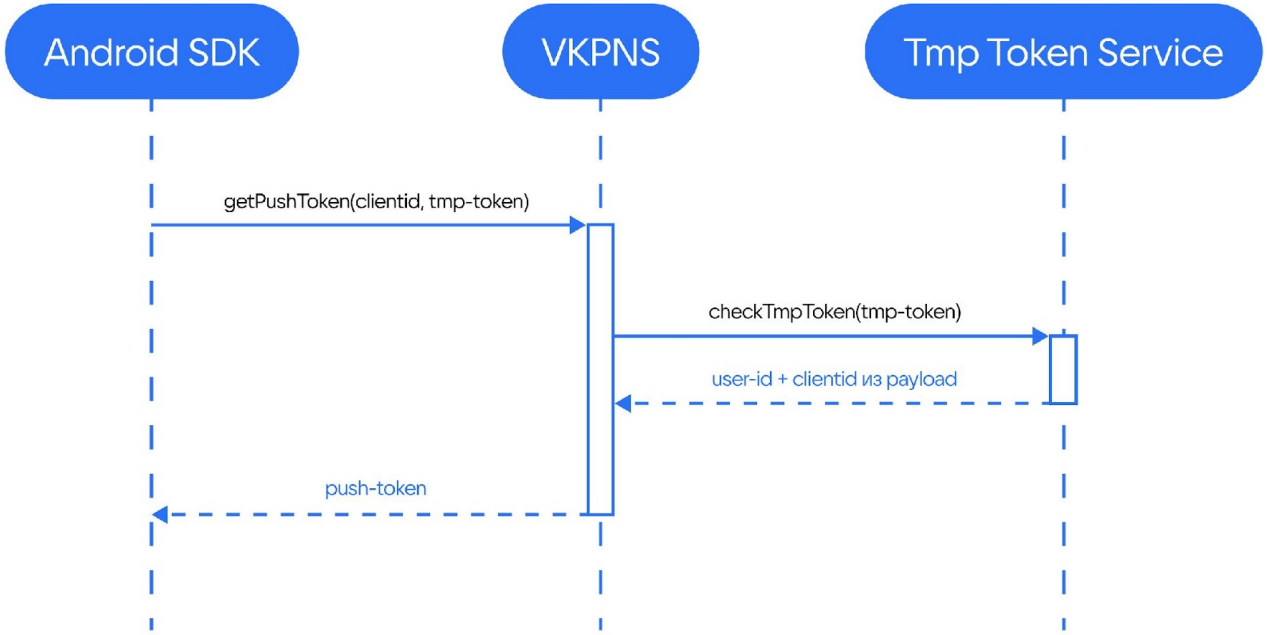
# DoS protection on generation of push tokens





# DoS protection
# on generation of push tokens





RuStore

# How we integrated RuStore push notifications into Mail.ru Mail Service

# Integration process to Mail.ru Mail Service

# Integration process to Mail.ru Mail Service



- Started sending push notifications through RuStore in parallel with FCM

# Integration process to Mail.ru Mail Service



- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour

# Integration process to Mail.ru Mail Service



- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics

# Integration process to Mail.ru Mail Service



- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics
- Will start showing the push that came before

# Integration process to Mail.ru Mail Service



- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics
- Will start showing the push that came before
- Will measure delivery latency

# Integration process to Mail.ru Mail Service



- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics
- Will start showing the push that came before
- Will measure delivery latency
- Will reduce delivery latency

# Complexity of integration

# Complexity of integration



- To enable push notifications, you need to make just a couple of clicks in the RuStore developer console

# Complexity of integration



- To enable push notifications, you need to make just a couple of clicks in the RuStore developer console
- The interface of our SDK repeats the interface of the FCM SDK

# Complexity of integration



- To enable push notifications, you need to make just a couple of clicks in the RuStore developer console
- The interface of our SDK repeats the interface of the FCM SDK
- The push notification sending API is a drop-in replacement for the FCM API

# Benefits of integration

# Benefits of integration



- Protected in case Google blocks our pushes

# Benefits of integration



- Protected in case Google blocks our pushes
- We can increase the delivery rate by sending via two channels in parallel

# Benefits of integration



- Protected in case Google blocks our pushes
- We can increase the delivery rate by sending via two channels in parallel
- Carried out load testing of RuStore push notifications in real conditions

# Midterm results (VKPNS vs FCM)

# Midterm results (VKPNS vs FCM)



- Number of successfully sent push notifications per day: 99.999% vs 99.993%

# Midterm results (VKPNS vs FCM)



- Number of successfully sent push notifications per day: 99.999% vs 99.993%
- API send average response time: 13ms vs 45ms

# Midterm results (VKPNS vs FCM)



- Number of successfully sent push notifications per day: 99.999% vs 99.993%
- API send average response time: 13ms vs 45ms
- Push notification deliverability per day: ~101% vs 100%

# Future plans

# Future plans



- Get closer in functionality to FCM (broadcasting by topics, more push settings, show deliverability metrics)

# Future plans



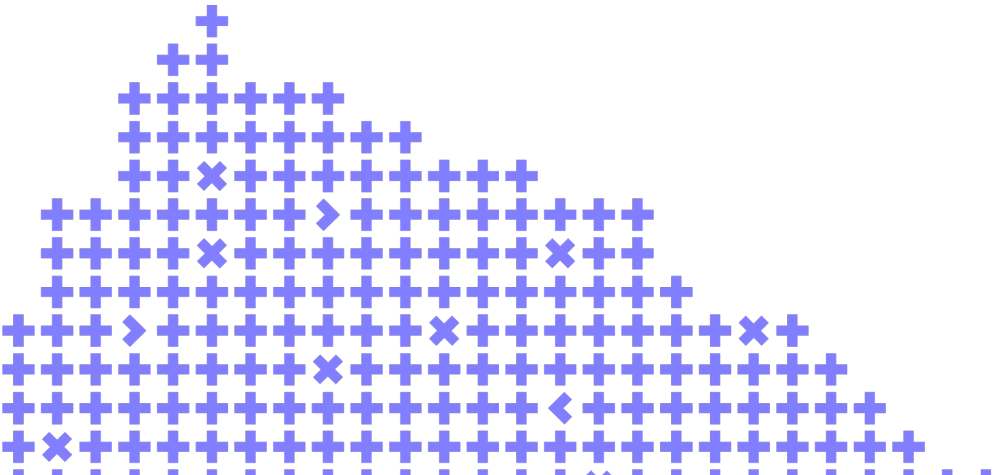
- Get closer in functionality to FCM (broadcasting by topics, more push settings, show deliverability metrics)
- Start receiving push notifications through other VK applications on the user's device

# Future plans



- Get closer in functionality to FCM (broadcasting by topics, more push settings, show deliverability metrics)
- Start receiving push notifications through other VK applications on the user's device
- Maintain health metrics no worse than that of Google FCM

# Leave your feedback!




Contacts:
kirill.alekseev@corp.mail.ru


Slides
https://vk.cc/cjCU20

High Load Armenia

Co-organizer

Yandex